$ AF/2700

#19

Patent

Attorney's Docket No. P2380-505

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	)	**MS Appeal Brief-Patents**
David G. OPSTAD et al	)	Group Art Unit: 2672
Application No.: 09/306,888	)	Examiner: T. Havan
Filed: May 7, 1999	)	Appeal No.: Unassigned
For: AUTOMATIC SYNTHESIS OF FONT TABLES FOR CHARACTER LAYOUT	)	

Please Enter AF 9/8/03

BRIEF FOR APPELLANT

Commissioner for Patents
Alexandria, Virginia 22313-1450

Sir:

This appeal is from the decision of the Primary Examiner dated January 27, 2003 (Paper No. 17), finally rejecting claims 1-9, 11-13, 16-20, 22-27 and 29-31, which are reproduced as an Appendix to this brief.

A Government fee of $320.00 was paid when Appellants filed a Brief for Appellant on May 22, 2002, after which prosecution was reopened. Therefore, it is believed that no fee is due for filing the present Brief.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§1.16, 1.17, and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800. This paper is submitted in triplicate.

TABLE OF CONTENTS

I. Real Party in Interest

The application involved in this appeal, and the invention to which it is directed, are assigned in their entirety to Apple Computer Inc., of Cupertino, California, which is the real party in interest.

II. Related Appeals and Interferences

There are no other known appeals or interferences that will directly affect, or be affected by, or have a bearing on the Board's decision in this appeal.

III. Status of Claims

The present application contains claims 1-31.

Claims 1-9, 11-13, 16-20, 22-27 and 29-31 stand finally rejected.

Claims 10, 14, 15, 21 and 28 stand objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all the limitations of the base claim and any intervening claims.

IV. Status of Amendments

There were no amendments filed subsequent to the final Office Action.

V. Summary of the Invention

The claimed invention is directed to fonts that are employed in computer systems to display and/or print images of characters. In essence, a font comprises a number of data tables that relate to various parameters and other features that determine the appearance and "behavior" of the characters in a font as they are being displayed or printed. For example, one table may contain the various images, or glyphs, that are associated with each character. Other tables may contain data such as kerning, i.e., the spacing between characters, the metrics or dimensions of glyphs, variable properties such as line widths, and the like. The information provided by these tables is employed for a number of different purposes. For instance, the data in some of the tables is employed when a line of characters is to be laid out for display or printing purposes. (Page 1, line 24 to page 2, line 5.)

With reference to Figure 2, when a user types a character on a computer's keyboard 24, a call is made to the computer's imaging system 38 to display the character corresponding to the keystroke. The process that is undertaken to display the character is described in the application at page 6, line 14 to page 8, line 15. As a string of characters are supplied to the imaging system, it may call upon a line layout processor. This processor adjusts the position of individual glyphs relative to one another, and performs further modifications of the glyphs to assemble a complete line of characters. An example of the general procedures that are performed in the line layout processor is illustrated in Figure 4. As depicted therein, each step employs data contained in the tables of the font to perform the necessary modification of the glyphs. (Page 8, line 16 to page 10, line 20.)

The present invention is particularly concerned with the possibility that a given font may not contain all of the tables necessary for the imaging system, e.g. the layout processor, to perform all of the operations on the glyphs. For instance, an older font that was created before the capabilities of the layout processor were developed may not contain all of the requisite tables. In accordance with the invention, tables that are necessary in order to perform a certain operation on a font are automatically synthesized if they are missing from the original definition of the font. Figure 7 schematically depicts a process by which the table is automatically synthesized. The synthesizer employs data contained in some of the tables of a font, such as a glyph mapping table 47, to construct a font map 81. This font map provides certain characterizing information about each glyph in the font. For instance, in the example of Figure 7, the data for each glyph includes a glyph number, an identification value, properties and a unicode value. A mapping table 84 within the synthesizer defines relationships between characters. In a simplistic example depicted in Figure 7, the mapping table identifies the relationship of uppercase and lowercase characters. Based upon the information in this mapping table and the font map 81, the synthesizer creates a new table 86. Each entry in this table maps specific glyphs in the font to one another, in accordance with their uppercase/lowercase relationship. (Page 14, line 5 to page 16, line 11.)

Preferably, once a table is synthesized for a font, it is stored in a persistent manner so that its data is available for subsequent uses of the font. In accordance with another feature of the invention, the synthesized table is not stored as part of the original font definition. For instance, as illustrated in Figure 5, the original data for a font is contained within a suitcase 46. Automatically synthesized tables are stored in a separate annex 50 that is associated with the font. This annex is a file that contains additional information about the font, but does not affect the original font definition stored in the suitcase. (Page 12, line 15 to page 13, line 9.)

As a result of the capabilities provided by the present invention, it becomes possible for older fonts, or "lightweight" fonts having a limited number of data tables, to be employed in the context of newer display technologies that perform more complex manipulation of the character images. The basic operation of the invention within this context is depicted in the flow chart of Figure 6. When a particular line layout procedure is to be performed, a request for the tables that are pertinent to that procedure is made at step 60. A determination is then made at step 62 whether the requisite tables are present. If so, the procedure is carried out at step 64. If a table is not present, a determination is made at step 70 whether an annex exists. If so, it is checked for the presence of the table. If the table does not exist in the annex, it is automatically synthesized in accordance with the invention, and then placed in the annex. The synthesized table is then used by the procedure to carry out the necessary operations on the glyphs of the font. (Page 13, line 10 to page 14, line 4.)

VI. The Issues

The final Office Action presents a single issue for review on this appeal, namely whether claims 1-9, 11-13, 16-20, 22-27 and 29-31 are unpatentable under 35 U.S.C. § 103 over the *Sonnenschein* patent (U.S. Patent No. 5,500,931) in view of the *Patel et al.* patent (U.S. Patent No. 6,426,751).

VII. Grouping of Claims

Although all pending claims have been grouped in a single ground of rejection, not all of the claims stand or fall together. Rather, various ones of the claims present separate issues of patentability that must be considered independently of other claims. The separate bases of patentability, and the claims corresponding thereto, are presented in the arguments that follow.

VIII. Argument

A. The Rejection Does not Establish a *Prima Facie* Case of Obviousness

The Manual of Patent Examining Procedure, at Section 2143, sets forth three basic criteria for a *prima facie* case of obviousness. At issue on this appeal is the third of those criteria, namely "[t]o establish prima facie obviousness of a claimed invention all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974)." MPEP § 2143.03 (emphasis added). The final rejection merely discusses the claim limitations in general terms, with broad, vague references to the *Sonnenschein* and *Patel et al.* patents. It fails to identify how the references teach or suggest a number of the specific features of the claims.

The differences between the claims and the applied references become apparent upon a detailed analysis of representative claims. Before proceeding with an analysis, Appellant provides the following summary of the *Sonnenschein* and *Patel et al.* patents.

1. The *Sonnenschein* Patent

The *Sonnenschein* patent is concerned with solving problems involving operating systems which overload character codes[1] in their code page architectures and fonts (see the *Sonnenschein* patent, column 3, lines, 60-67), or operating systems that can detect when a

[1] According to the *Sonnenschein* patent, a *character code* is a mapping, often presented in tabular form, which defines a one-to-one correspondence between characters in a character repertoire (i.e., a set of distinct characters) and a set of nonnegative integers. That is, it assigns a unique numerical code, a code position, to each character in the repertoire. (See column 1, line 41).

character is missing, but would display each missing character as one particular glyph[2] (e.g., the "sigma" character). (*Id.*, at column 2, lines 18-22; and column 4, lines 27-31).

As described in the *Sonnenschein* patent, incorrect font application can result from "overloading" a same character code (i.e., representing characters in different scripts[3] with the same code). *Sonnenschein* discloses that systems having overloaded character codes cannot adequately detect, when a character is entered, whether that character exists in a currently used font. Hence, overloading can often cause an operating system to associate a desired character (e.g., a character of a particular font intended by an author or editor of text) with an entirely different character (or none at all). The *Sonnenschein* patent also discloses that overloading of character codes can cause "garbling" of text when a client tries to apply a font style to a string of characters of different fonts.

To solve these problems, the *Sonnenschein* patent discloses two sets of processes used by the text subsystem to manipulate character codes and font styles. The first process is used when a character, or a string of characters, is entered into a text stream and adds font styles to characters upon text entry. The second process applies font styles to selected text (e.g., when a user applies a font style change to a range of characters containing characters from multiple scripts).

In the first process, logic "automatically styles a character with a <u>new font style</u>[4] when the current font cannot display the character." (See *Sonnenschein*, column 5, lines 1-3 (emphasis added).) More specifically, the first process iterates through characters of a text string and, at each iteration, determines whether the current character code can be

[2] The *Sonnenschein* patent defines a *glyph* as a visual representation of a character code. (See column 2, lines 21-22.)

[3] The *Sonnenschein* patent defines a *script* as "a collection of character codes that have meanings that are semantically related. Usually a language maps to a single script." (See column 1, lines 45-47.)

[4] *Sonnenschein* describes a *font style* as "a symbolic reference to a font. All the functionality of a font is available through a font style." (See column 4, lines 35-37.)

mapped[5] to a valid glyph of a font. In the process, the character code in the current iteration is first checked to see whether it can be mapped to a valid glyph in a currently set font. If so, the current character is styled with the font. If the character code cannot be mapped to a valid glyph in the current font, the process then attempts to associate the current character code with a glyph in the font associated with the current character (because "[e]very character in a text stream has a font style associated with it," see *Sonnenschein*, column 5, lines 27-28). Next, if the first and second processes fail, all available fonts in the system are checked to find a font having a valid glyph. Finally, if the current character cannot be associated with a valid glyph of any of the system fonts, it is associated with a "predefined missing glyph system font," which is described in *Sonnenschein* as "a special font that is used for character substitution purposes only when no other font can display the character." (See *Sonnenschein*, column 5, lines 31-34 and column 6, lines 3-5.)

The second process is described with reference to Figure 9 and column 8, lines 19-37 of the *Sonnenschein* patent. In the second process, a font style change is applied to a range of characters by iterating through each character in the range of characters. For each iteration, a decision is made as to whether a current character is in the font to be applied. If so, the character is styled with that font. Otherwise, the character font style remains unchanged and the process proceeds on to the next iteration.

2. The *Patel et al.* Patent

The *Patel et al.* patent is directed to a method, system and computer program product that a font editor can use to either add typographical features to an existing font file or create a new font file. The method first includes creating a front-end editable text file called "a feature file" in which a user writes logical statements using a high-level feature

[5] It is noted that *Sonnenschein* describes "mapping" a character code to a valid glyph. (See the *Sonnenschein* patent, column 5, line 48 to column 6, line 11.) *Mapping* in the *Sonnenschein* patent pertains to determining whether a character is available for a particular character code in a font's character mapping table. (See Figure 2A, item 240, and column 5, lines 51-54.)

definition language for the specification of various typographical features, such as layout features, that may enhance or supplement a source font. (See column 3, lines 16-28.) The feature file is read and parsed to create internal representations of the feature file statements. (See column 3, lines 29-34.) The program translates the internal representations and creates actual subtables and other data, which is stored in the font to define changes to the font (see column 4, lines 52-55) or convert a input font, such as a Type 1 font, into an output font, such as and OpenType font (see column 5, lines 9-23).

3. Claims 1, 11, 19 and 26

Turning to representative claim 1, this claim is directed to a method for generating an image of a sequence of characters which comprises, *inter alia*, the steps of "*retrieving glyphs from a font which correspond to characters in a string of characters*" and "*determining whether the font contains a predetermined data table that pertains to the layout of the glyphs.*" Claim 1 further requires that *if the font is determined not to contain the data table, the table is automatically synthesized based upon data contained in the font.*

In contrast, the *Sonnenschein* patent does not teach or suggest synthesizing a data table pertaining to the layout of a glyph. Rather, *Sonnenschein* teaches that each character of inserted text is analyzed, one-by-one, to determine if a current character is in a last font processed set, and if it is not, to try to style the character with another font available to the system (including the possible assignment of a "missing glyph system font when a valid font cannot otherwise be found, see column 6, lines 3-5). (See *Sonnenschein*, column 5, lines 48-51.) Hence, *Sonnenschein* looks only at fonts which already exist in the system to determine whether a character code of an inserted character matches one of these fonts, and if a match is not determined for a particular font, the process then proceeds to check another font available in the system. The *Sonnenschein* patent is not concerned with creating any tables for a font because it always looks for other fonts to represent a character if it is found that a current font cannot represent that character. The *Sonnenschein* patent therefore does not teach or suggest anything with regard to the synthesis of a data table of a font if it is found that the data table does not exist, as set forth in claim 1.

With respect to the step of "determining whether the font contains a predetermined data table that pertains to the layout of glyphs," as recited in claim 1, the final Office Action does not provide any support for this feature in the *Sonnenschein* patent.[6] Moreover, the *Sonnenschein* patent does not disclose anything in particular with respect to *data tables* that pertain to the "layout" of glyphs, e.g., positioning them relative to one another to form a line of characters. As mentioned above, the *Sonnenschein* patent looks for a current character in the last font processed, in the current character's own font style, or in another available font in a list of fonts that are available in the system.

It should be noted that with respect to independent claims 1, 11, 19 and 26, the final Office Action, at page 4, lines 7-8, assets that the *Sonnenschein* patent teaches "data tables that contain information about glyphs in the font (figs. 4-6)." Figures 4-6 of *Sonnenschein* do not, however, show *predetermined data tables* of a font. These figures are described in *Sonnenschein* as examples that "further illuminate" how characters would be mapped to various fonts already existing in the system. (See column 6, lines 35-62.) That is, they merely serve to illustrate a concept, and do not constitute a data structure from which a new table is synthesized. There is simply no teaching or suggestion whatsoever in these conceptual examples of automatically synthesizing a data table pertaining to the implementation of a font when such table is determined to be missing in a font, as recited in the context of claim 1.

In any event, whether or not any tables can be inferred from *Sonnenschein* that pertain to the layouts of glyphs, the referenced portion of the patent does not disclose, nor otherwise suggest, the step of *determining* whether a font contains a predetermined data table. Rather, the procedure of the *Sonnenschein* patent operates on the basis that all necessary tables of a font are present, and it simply determines whether one of these fonts includes a valid glyph for a character code. In *Sonnenschein*, if a character code cannot be mapped to a font, the processor looks among other fonts in the system for a match and

---

[6] In rejecting claims 1-5 and 9, the final Office Action refers to the statements made in connection with the rejection of claims 11-13, 16-20 and 22-27. (See page 5, lines 20-21.)

replaces the font associated with the current character with the newly found one. In other words, it discloses techniques for styling a character with a another existing font style when a current font cannot display the character. It does not disclose any mechanism for automatically synthesizing data *determined* to be missing from the font.

The significance of this determination is brought out in the step of "automatically synthesizing said data table, based upon data contained in the font, *if* the font is determined not to contain said data table" recited in claim 1. Thus, in the context of claim 1, the automatic synthesis of a data table is conditional upon its presence or absence in a font. A determination is first made *whether* the font contains a predetermined data table. Then, *if* the condition should occur that the table is not present, it is automatically synthesized from data contained in the font. Thereafter, the glyphs are laid out in a line, in accordance with the data in the automatically synthesized table, and an image of the laid-out line of glyphs is generated.

The procedure recited in claim 1 operates to create a table for a font *if* that table is not already present in the font. The *Sonnenschein* patent does not disclose any such conditionality based upon the presence or absence of a table in a font. In particular, there is no teaching to suggest the step of determining whether the base font contains a predetermined data table. Rather, because the *Sonnenschein* patent is concerned with a totally different type of operation, it operates on the assumption that all necessary tables are present.

Furthermore, the *Sonnenschein* patent does not disclose any procedure for automatically synthesizing a missing data table, if one were to be absent. With respect to this point, the final Office Action states that *Sonnenschein* teaches "a font table synthesizer which is responsive to the absence of a predetermined data table for creating and storing [sic, a] table on the basis of data contained in the font file (col. 4, lines 41-52)." The undersigned has reviewed this portion of the *Sonnenschein* patent and cannot find any support whatsoever for this allegation. In fact, the cited portion of *Sonnenschein* discloses the following:

> The first process is used when a character, or string of characters, is entered into a text string. If the character is missing from the font specified in the current typing style the process will choose a font that can display the character. For example, if the code for a "Σ" is inserted before "n/2" the process would automatically style the code with a font that could display the "Σ."
>
> The second process is used when a client applies a font style change to a range of characters. The process intelligently applies the font to the selection. For example, applying the preferred embodiment of the invention, Chicago font to the characters "Σ n/2" produce "Σ n/2" (emphasis added).

Appellants assert that this portion of the *Sonnenschein* patent does not disclose that if one or more required tables is missing, it can be automatically synthesized from data contained in the font file. As pointed out above, the *Sonnenschein* patent assumes that all of the necessary tables for generating a font will be present. Thus, with reference to claim 1, the *Sonnenschein* patent does not disclose the step of "determining *whether* the font contains a predetermined data table ..." Similarly, it does not disclose the following step of "automatically synthesizing said data table ... *if* the font is determined not to contain said data." As set forth in this claim, the present invention operates to *detect* whether the tables necessary to lay out glyphs are present in a font, and then automatically *creates*, or *synthesizes*, a table if it is missing. The *Sonnenschein* patent does not disclose these operations, nor anything analogous thereto. In particular, there is no disclosure of the capability to determine that a required table is missing, and thereafter synthesize the table in response to such a determination.

A significant distinction here is the fact that the procedure of the present invention can be carried out within the context of a single font. For instance, claim 1 recites the steps of "retrieving glyphs from *a* font ...", "determining whether *the* font contains a predetermined data table ...", and "automatically synthesizing *said* data table, based upon data contained in *the* font ..." In other words, information from a font that is being employed to generate images is used for the creation of a data table missing from *that* font. In contrast, the *Sonnenschein* patent does not disclose the generation of a data table that is missing from a font. Rather, the *Sonnenschein* patent discloses procedures for finding a font that includes a valid glyph for a character in inserted text. Thus, while the

Sonnenschein patent discloses a technique for automatically locating a font that can display a valid glyph corresponding to a character of inserted or selected text, the result is significantly different from the data table that is constructed in accordance with the present invention. In the present invention, a new set of data is constructed for an existing font. In the *Sonnenschein* patent processes, the data of fonts remain the same.

These same distinctions apply to claims 11, 19 and 26, as well as their dependent claims. Claim 11 recites that the font table synthesizer is "*responsive to the absence of a predetermined data table.*" Claims 19 and 26 recite that the data table is synthesized "*if the table is not present in the font file.*" These features are not taught in the *Sonnenschein* patent.

The Office Action acknowledges that the *Sonnenschein* patent fails to disclose creating a table. (See the final Office Action, page 4, line 11.) To make up for this shortcoming, the final Office Action proposes modifying the system of the *Sonnenschein* patent with a teaching in the *Patel et al.* patent of creating a table for font layout. Appellants respectfully submit, however, that the proposed combination fails to establish a *prima facie* case of obviousness because these patents do not teach or suggest the combination of every recited feature in each of the claims.

Patel et al. is directed to a computer program having a front-end editable text file called "a feature file" that a user can edit and have a computer process to define changes to an existing font file or to create a font file. *Patel et al.* describes the feature file as being an editable file that includes simple logical statements that can be set by a font editor in the form of high-level feature definition language, for example, in logic statements expressed in "English-like grammar." (See *Patel et al.*, column 3, lines 22-28.)

The Office Action alleges that *Patel et al.* teaches constructing a table that "contains information on glyph positioning, glyph substitution, justification, and baseline positioning," and that by "taking the combined teaching of *Sonnenschein* and *Patel* as a whole, it would have been obvious to combine the teaching of *Patel* to the system of *Sonnenschein* because doing so would have enabled processing fonts to improve font layout in a table format as noted in *Patel* (col. 1, lines 17-41; figs. 2 and 4-element 420) [sic]" (emphasis in original). Appellants respectfully submit, however, that even if one of ordinary skill in the art was led to combine the teachings of the *Sonnenschein* and *Patel et al.* patents, such combination would not have resulted in the claimed invention.

The textual part of *Patel et al.* cited in the final Office Action (i.e., column 1, lines 17-41) is not directed to *synthesizing* a table in the context in which this feature is recited in the pending claims. Rather, the cited portion of *Patel et al.* relied upon by the Office describes conventional OpenType font design in which data tables are generated by specific programs written to generate binary data, or by preparing a text input file that details the values that go into each font table data structure and then assembling the text into binary form. The *Patel et al.* patent is directed to the latter of these methods. Specifically, the *Patel et al.* patent teaches a feature file that is composed and processed by a font editor to alter an existing font file or to create a new font file. However, there is no mention or suggestion in *Patel et al.* that creation of the disclosed data tables are carried out as a result of a condition of determining that a font does not contain a predetermined data table, as recited in claim 1. Similarly, *Patel et al.* does not teach *a font table synthesizer* which is *responsive to the absence of a predetermined data table* for creating and storing said table, as recited in claim 11, or *determining whether the data table is present in a file containing a font,* and *synthesizing said table* from data contained in said file *if the table is not present in the file*, as recited in claims 19 and 26. Rather, modification of fonts in the context of the *Patel et al.* patent pertains to processes initiated by a font designer or editor that begin with creating a feature file.

According to *Patel et al.*, when a feature file is read and parsed, the process of table creation is defined by "target and replacement GNodes." (See column 5, lines 7-8.) GNodes, in turn, are defined for each glyph in every feature file rule. (See column 3, lines 51-52.) *Patel et al.* does not mention any step of *determining* whether a font contains particular data table. Moreover, because there is no teaching or suggestion in *Patel et al.* of this determination, it necessarily follows that *Patel et al.* does not teach the step of automatically creating any data table that is determined not to be contained in the font file. Hence, *Patel et al.* does not teach *retrieving glyphs from a font which correspond to* *characters in a string of characters, determining whether the font contains a predetermined data table that pertains to the layout of the glyphs, and automatically synthesizing said data table, based upon data contained in the font, if the font is determined not to contain said data table* ...," as recited in claim 1.

The suggested combination of the *Sonnenschein* and *Patel et al.* patents would perhaps have lead to a system that included one or more available fonts that were modified or created using a feature file, as taught in the *Patel et al.* patent, and included the ability to attempt a mapping, for each character inserted into a text string, to a valid glyph of one of the fonts available to the system (which would have possibly included the modified or created fonts). When comparing claim 1 with this hypothetical combination, it is clear that *Sonnenschein* and *Patel et al.* do not teach a system that includes the step of "*determining whether the font contains a predetermined data table that pertains to the layout of glyphs ... automatically synthesizing said data table, based upon data contained in the font, if the font is determined not to contain said data table.*" Because the proposed combination does not teach all of these features, claim 1 is patentable at least because a *prima face* case of obviousness does not exist. Other distinguishing features defined in independent claims 11, 19 and 26 were noted above. Appellants submit that these claims similarly recite features neither taught nor suggested the *Sonnenschein* and *Patel et al.* patents, whether considered individually or in any combination. As such, claims 11, 19 and 26 also are patentable.

4. Claims 6, 16, 22 and 29

As another point of distinction, the particular manner in which a data table is constructed in a preferred embodiment of the present invention is significantly different from the processes of the *Sonnenschein* and *Patel et al.* patents. For example, claim 6 recites the steps of "*building a font map that contains information about individual glyphs in the font,*" "*determining relationships between items of information in the font map,*" and "*constructing a table which identifies said relationships.*" Similar recitations are found in claims 16, 22 and 29. With respect to this subject matter, the final Office Action alleges that the *Sonnenschein* patent, at column 5, lines 12-27, discloses "a method for automatically synthesizing a data table that contains information about glyphs in a font, comprising the steps of building a font map that contains information about individual glyphs in the font ...." (See the final Office Action, page 5, lines 4-8). However, the cited portion of *Sonnenschein* relied on for allegedly teaching "*building font map* ..." actually sets forth the following:

> The logic set forth in FIG. 2 automatically styles a character with a new font style when the current font cannot display the character. The process iterates on the input text to determine if the font associated with the current character can display that character. If not, it tries to find a font that can. The process then tries to keep all subsequent characters of the input text in the same font.
>
> With reference to FIG. 3, to find a font that can display the current character, the process parses all the previous characters in the text stream and then parses all subsequent characters until a font referenced by one of those characters can display the current character. If no font is found, the system searches all the available fonts on the system (in a sorted order either specified by a user preference or by a stylistic matching process).

It is not apparent what information disclosed in this portion of the patent is considered to constitute *building a font map* as claimed. As noted above, the *Sonnenschein* patent determines whether a character of a current iteration can be mapped to a valid glyph of a font available to the system. This is accomplished in *Sonnenschein* by checking an existing character mapping table of each font of a list of available fonts (see *Sonnenschein*, Figure 2, item 240) and repeating this for each font in the list until a mapping to a valid glyph is found (see *Sonnenschein*, Figure 3, item 380). The *Sonnenschein* patent, however, does not teach *building a font map* as claimed. As pointed out above, *Sonnenschein* does not appear to alter fonts in any way, but instead merely looks to find another font that can display the character.

The *Sonnenschein* patent further does not teach *determining the relationships between items of information in a font map*, and, as correctly acknowledged in the final Office Action, "*constructing a table which identifies said relationships*." The final Office Action asserts that the *Patel et al.* patent teaches constructing a table for the font layout. The *Patel et al.* patent, however, does not remedy the deficiencies noted above of the

Sonnenschein patent because *Patel et al.* also does not mention or suggest *building a font map* as claimed. Hence, a *prima facie* case of obviousness has not been established for claims 6, 16, 22 and 29, and hence claims depending therefrom, because the applied references fail to teach all of the claimed features.

5. Claims 7, 8, 17, 18, 23, 24, 30 and 31

It is respectfully submitted that claims 7 and 8 are allowable at least for the reasons given above for claim 1, and further for the additional features recited. For example, claims 7 and 8 recite further features of the invention relating to the manner in which the font table is synthesized. In particular, claim 7 recites that some of the information in the font map is specific to the font, whereas other information is generic to multiple fonts. Claim 8 recites that the synthesized table contains font-specific information that is determined with reference to generic information. Claims 17, 23, 30 and 18, 24, 31 recite similar subject matter as claims 7 and 8, respectively. The final Office Action alleges that these features are disclosed in column 5, lines 39-59 of the *Sonnenschein* patent. However, it is not apparent what relevance *Sonnenschein's* disclosure of a loop process for finding a font with a valid glyph for a current character has to the claimed font map including font specific and font generic information. It is respectfully submitted that the *Sonnenschein* patent does not teach these claimed features.

6. Claims 4, 5, 13, 20 and 27

Each of dependent claims 4, 5, 13, 20 and 27 depend from one of claims 1 , 11, 19 and 26, and are therefore allowable at least for the above reasons. These dependent claims recite combinations including additional features that are not taught or suggested in *Sonnenschein* or *Patel et al.* For example, claim 4 recites that the synthesized data table is stored in a persistent annex file that is associated with, but separate from, the font. Claims 13, 20 and 27 recite similar subject matter. Claim 5 recites that a determination is made whether a missing data table is stored in an annex file. In the rejection of these claims, the final Office Action analogizes the annex file to a portion of the specification for the feature file, as set forth in columns 7 to 14 (i.e., Appendix A) of the *Patel et al.* patent. The undersigned has reviewed columns 7 to 14 of the *Patel et al.* patent cited by the Office Action for allegedly teaching these features, but could not find support for the features recited in claims 4, 5, 13, 20 and 27.

Appendix A of the *Patel et al.* patent contains specifications for the high level feature definition language used in a feature file. (See column 3, lines 20-22.). As described in column 3, line 10 to column 4, line 51, a feature file is a front end editable text file in which a user of a computer system can define changes to an existing file, or to create a font file. A feature file is processed to create subtables and other output data that is stored in a font file being changed or in a new file of a font being created. The feature file, however, is not the same as the annex file recited in the claims at least because a feature file is neither a persistent file nor an annex file in which a synthesized data table is stored. Moreover, any table created as a result of processing a feature file is not stored in a feature file, and is instead stored either in the font file or in a file defining a new font. In contrast, the annex file of the present invention contains *additional* synthesized data tables beyond those which form the original font definition. The two types of files serve different purposes, and are used in substantially different manners.

Furthermore, with respect to claim 5, there is no disclosure in the *Sonnenschein* and *Patel et al.* patents that a determination is made whether a data table is stored in an annex file as part of the process for determining whether the data table needs to be synthesized. As pointed out above, the system of the *Sonnenschein* patent does not check font files to see *whether* they contain a particular data file. In contrast, it merely checks whether the character code maps to a valid glyph in an existing table. Nor does the *Patel et al.* patent teach determining whether a particular data table is stored in an annex file associated with a font and carrying out the synthesis of the data table only if the data table is not contained in either the font or the annex file.

7. Claims 2, 3, 9, 12 and 25

For at least the same reasons discussed above, Appellants respectfully request that the rejections of claims 2 and 3 be reversed. Because the *Sonnenschein* and *Patel et al.* patents do not teach or suggest determining whether a font contains a predetermined data table that pertains to the layout of glyphs and automatically synthesizing said data table, based upon data contained in the font, if the font is determined not to contain said data table, claims 2 and 3 further define novel and inventive subject matter.

With respect to claims 9 and 25, Appellants respectfully request that the rejection of this claim be reversed, if for no other reason than because these dependent claims include all the limitations of claims 1 and 19, respectively, which are patentable for reasons are set forth above regarding the novel and inventive nature of both these claims. Furthermore, each of claims 9 and 25 recite the additional feature of synthesizing a data table for a font file by retrieving data from the font and storing the retrieved data in a table having a predetermined format. Although both of these claims have been rejected, the final Office Action does not provide any support in the *Sonnenschein* or *Patel et al.* patents for their rejection. Without such a showing, the rejection is not supportable.

Claim 12 recites that *a font subsystem determines whether a predetermined data table is contained in either the font file, and causes the synthesizer to create the table when a determination is made that the table is not present in the font file*. The final Office Action alleges that the *Patel et al.* patent teaches these features in column 4, line 52 to column 5, line 7 and Figure 4. However, for the reasons set forth above with respect to claim 11, neither the *Sonnenschein* patent nor the *Patel et al.* patent teach determining whether a predetermined data table is absent in a font file. Hence, the further features of a font subsystem which causes a synthesizer to create a table based on its determination that the table is contained in a font file, as recited in claim 12, defines further points of distinction not found in either of the applied patents. Because these features are not taught in the applied art, the rejection is improper because a *prima facie* case has not been established.

IX. Conclusion

From the foregoing, it can be seen that the *Sonnenschein* and *Patel et al.* patents fail to disclose, or otherwise suggest, a number of the claimed elements. Consequently, the Examiner has failed to establish a *prima facie* case of obviousness, as required for a rejection under 35 U.S.C. § 103.

The rejection of the claims is not properly founded in the statute, and should be reversed.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

By: [illegible]
John F. Guay
Registration No. 47,248

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

Date: June 26, 2003

APPENDIX A

The Appealed Claims

1. A method for generating an image of a sequence of characters, comprising the steps of:

retrieving glyphs from a font which correspond to characters in a string of characters;

determining whether the font contains a predetermined data table that pertains to the layout of glyphs;

automatically synthesizing said data table, based upon data contained in the font, if the font is determined not to contain said data table;

laying out the glyphs in a line, in accordance with the data in said table; and

generating an image of the laid-out line of glyphs.

2. The method of claim 1 wherein said generating step includes displaying the line of glyphs on a display device.

3. The method of claim 1 wherein said generating step includes printing the line of glyphs in a document.

4. The method of claim 1 further including the step of storing the synthesized data table in a persistent annex file that is associated with, but separate from, the font.

5. The method of claim 1 further including the step of determining whether said data table is stored in an annex file associated with the font, and wherein said automatic synthesis step is carried out only if the table is not contained in either the font or the annex file.

6. The method of claim 1 wherein the step of automatically synthesizing said data table comprises the steps of:

building a font map that contains information about individual glyphs in the font;

determining relationships between items of information in the font map; and

constructing a table which identifies said relationships.

7. The method of claim 6 wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

8. The method of claim 7 wherein the synthesized table contains font-specific information that is determined with reference to generic information.

9. The method of claim 1 wherein the step of automatically synthesizing said data table comprises the steps of retrieving data from the font and storing the retrieved data in a table having a predetermined data format.

10. The method of claim 1 further including the steps of determining whether said data table is of a first type or a second type when the data table is determined not to be

present in the font; directly initiating said synthesizing step if said data table is of said first type; or, providing an indication that said data table is not present in the font if said data table is of said second type, and initiating said synthesizing step upon receipt of a request that is responsive to said indication.

11. A system for generating images of characters, comprising:

a font subsystem which is responsive to identification of characters to access at least one font file to retrieve glyphs associated with the identified characters, and data tables that contain information about glyphs in the font; and

a font table synthesizer which is responsive to the absence of a predetermined data table for creating and storing said table on the basis of data contained in the font file.

12. The system of claim 11 wherein said font subsystem determines whether a predetermined data table is contained in the font file, and causes said synthesizer to create said table when a determination is made that the table is not present in the font file.

13. The system of claim 11 wherein said font synthesizer stores said table in an annex file that is associated with, but separate from, the font file.

14. The system of claim 13 wherein said font subsystem determines whether a predetermined data table is contained in either the font file or the annex file, and causes said synthesizer to create said table when a determination is made that the table is not present in either the font file or the annex file.

15. The system of claim 12 wherein said font subsystem operates in a first mode to cause said synthesizer to automatically create the table in response to said determination, and in a second mode to provide an indication when a data table is determined not to be present and thereafter cause said synthesizer to create the table in response to a request that is responsive to said indication.

16. A method for automatically synthesizing a data table that contains information about glyphs in a font, comprising the steps of:

building a font map that contains information about individual glyphs in the font;

determining relationships between items of information in the font map; and

constructing a table which identifies said relationships.

17. The method of claim 16 wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

18. The method of claim 17 wherein the synthesized table contains font-specific information that is determined with reference to generic information.

19. A method for providing data that relates to the implementation of a font, comprising the steps of:

receiving a request for a data table that pertains to the implementation of a font;

determining whether the data table is present in a file containing the font; and

synthesizing said table from data contained in said file if the table is not present in the font file.

20. The method of claim 19 further including the step of storing the synthesized table in an annex file separate from said font file.

21. The method of claim 19 further including the steps of determining whether said data table is of a first type or a second type when the data table is determined not to be present in the font file; automatically initiating said synthesizing step if said data table is of said first type; or, providing an indication that said data table is not present in the font file if said data table is of said second type, and initiating said synthesizing step upon receipt of a request that is responsive to said indication.

22. The method of claim 19 wherein the step of synthesizing said data table comprises the steps of:

building a font map that contains information about individual glyphs in the font;

determining relationships between items of information in the font map; and

constructing a table which identifies said relationships.

23. The method of claim 22 wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

24. The method of claim 23 wherein the synthesized table contains font-specific information that is determined with reference to generic information.

25. The method of claim 19 wherein the step of synthesizing said data table comprises the steps of retrieving data from the font and storing the retrieved data in a table having a predetermined data format.

26. A computer-readable medium containing a program which executes the steps of:

receiving a request for a data table that pertains to the implementation of a font;

determining whether the data table is present in a file containing the font; and

synthesizing said table from data contained in said file if the table is not present in the font file.

27. The computer-readable medium of claim 26, wherein said program executes the further step of storing the synthesized table in an annex file separate from said font file.

28. The computer-readable medium of claim 26, wherein said program executes the further step of determining whether said data table is of a first type or a second type when the data table is determined not to be present in the font file; automatically initiating said synthesizing step if said data table is of said first type; or, providing an indication that said data table is not present in the font file if said data table is of said second type, and

initiating said synthesizing step upon receipt of a request that is responsive to said indication.

29. A computer-readable medium containing a program which executes the steps of:

building a font map that contains information about individual glyphs in a font;

determining relationships between items of information in the font map; and

constructing a table which identifies said relationships.

30. The computer-readable medium of claim 29, wherein some of the information in said font map is specific to the font, and other information is generic to multiple fonts.

31. The computer-readable medium of claim 30, wherein the synthesized table contains font-specific information that is determined with reference to generic information.